Heap Integrity Detection

Daniel Plakosh, Software Engineering Institute [vita[1]]

2005-09-27 L1 / D/P, L[2]

This article describes a system to protect the glibc heap by making modifications to the chunk structure and management functions.

Development Context

Dynamic memory management

Technology Context

C, glibc, GCC, dlmalloc

Attacks

Attacker executes arbitrary code on machine with permissions of compromised process or changes the behavior of the program.

Risk

Standard C dynamic memory management functions such as `malloc()`, `calloc()`, `realloc()`, and `free()` [ISO/IEC 99] are prone to programmer mistakes that can lead to vulnerabilities resulting from buffer overflow in the heap, writing to already freed memory, and freeing the same memory multiple times (e.g., double-free vulnerabilities).

Description

Robertson and colleagues devised a system to protect the `glibc` heap by making modifications to the chunk structure and management functions [Robertson 03].

Figure 1. Modified memory chunk structure

```
1. struct malloc_chunk {
2.   INTERNAL_SIZE_T magic;
3.   INTERNAL_SIZE_T __pad0;
4.   INTERNAL_SIZE_T prev_size;
5.   INTERNAL_SIZE_T size;
6.   struct malloc_chunk *bk;
7.   struct malloc_chunk *fd;
8. };
```

This heap integrity scheme prepends a canary and padding field to the chunk structure as shown in Figure 1. The canary contains a checksum of the chunk header seeded with a random value. The global checksum seed value is stored in the `__heap_magic` `static` variable. This variable is initialized during process startup with a random value, which is then protected against further writes by `mprotect()`.[16]

The heap protection system also augments the heap management functions with code to manage and check each chunk's canary. The canary in a newly allocated chunk is initialized to a checksum that includes its memory location and size fields and is seeded with the global value of `__heap_magic`. When a chunk

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/268-BSI.html (Plakosh, Daniel)
16. The mprotect function modifies the access protection of a mapped file region or anonymous memory region created by the `mmap()` function.

is returned by a call to `free()`, the chunk's canary is checked against the checksum calculated when the chunk was allocated. If the checksums do not match, an exception is raised and the process is aborted.

References

[ISO/IEC 99]	ISO/IEC. *ISO/IEC 9899 Second edition 1999-12-01 Programming languages — C*. International Organization for Standardization, 1999.
[Robertson 03]	Robertson, William; Kruegel, Christopher; Mutz, Darren; & Valeur, Fredrik. "Run-time Detection of Heap-based Overflows," 51-60. *Proceedings of the 17th Large Installation Systems Administration Conference*. San Diego, CA, October 26–31, 2003. Berkeley, CA: USENIX Association, 2003.

Pearson Education, Inc. Copyright